OAuth for Web Server Applications

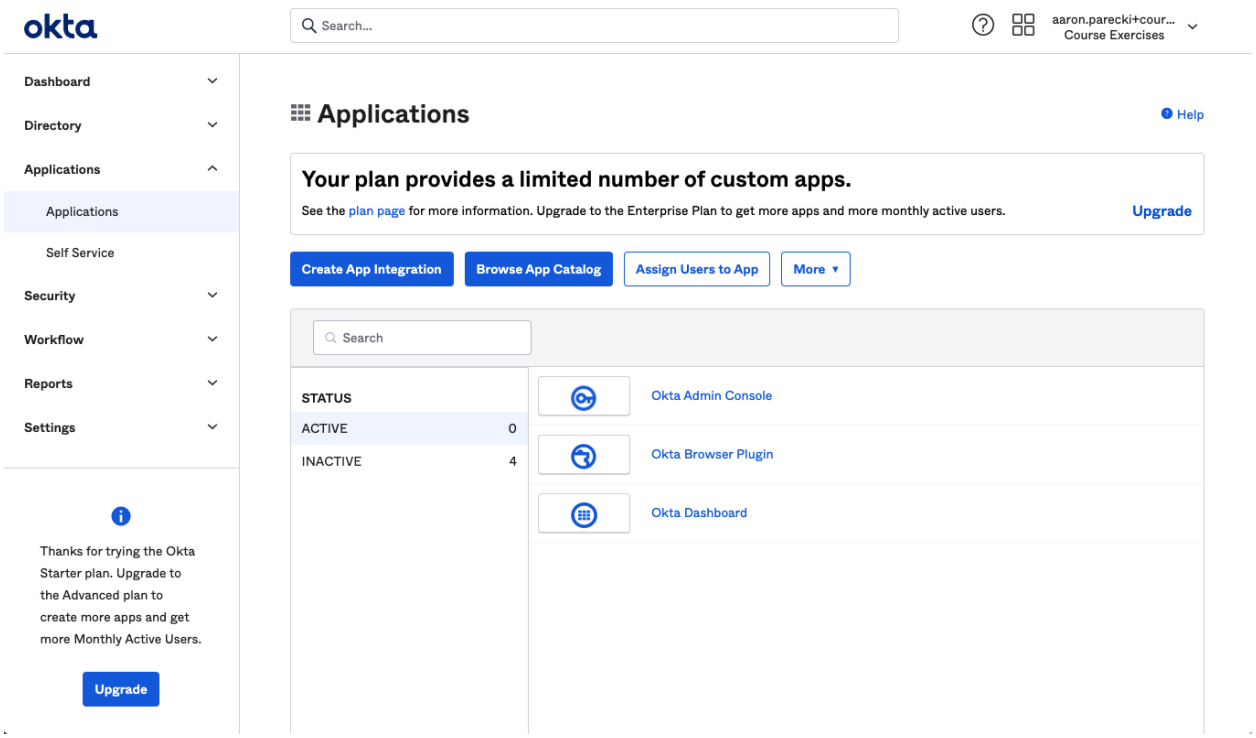
Description
This exercise will walk you through the OAuth flow you would use if you are building a web server based application. The goal of this exercise is to use the authorization code flow to get an access token.

Estimated Duration
15 minutes

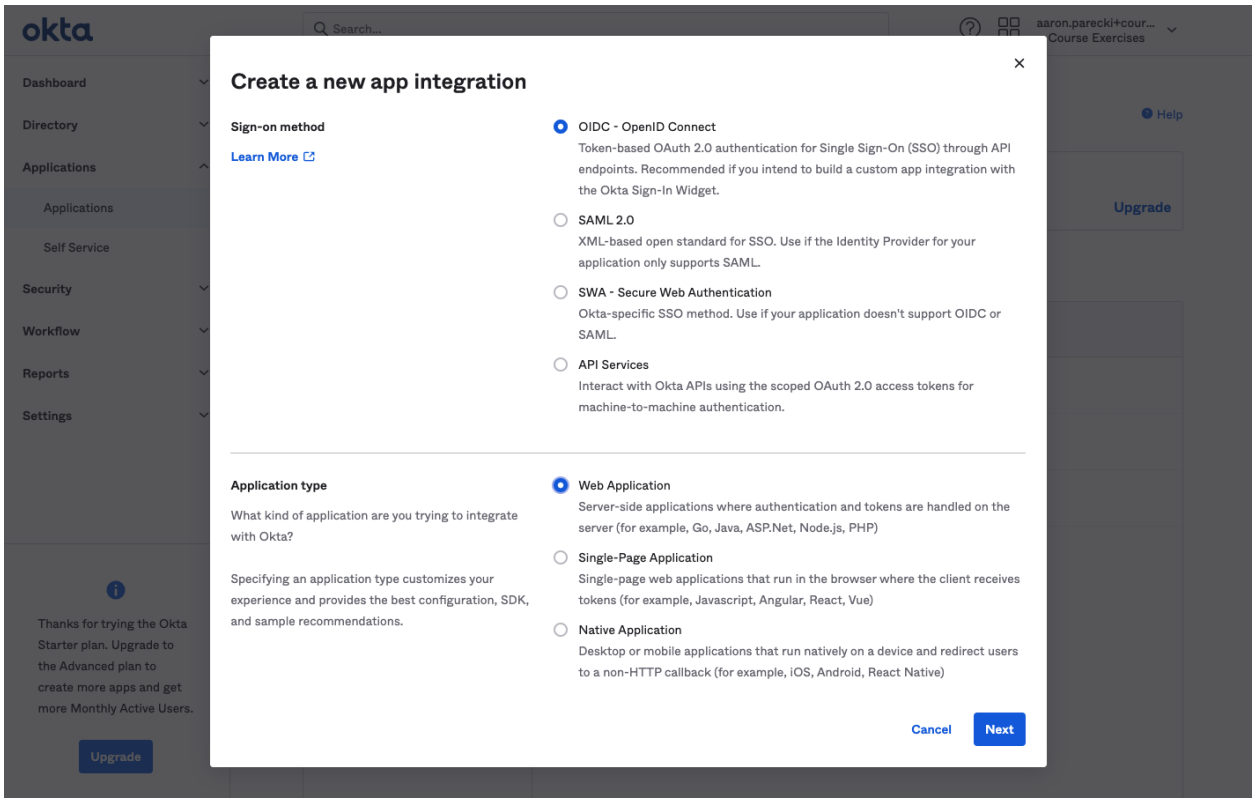
Instructions
Make sure you've completed the first Getting Started exercise, as you'll need the account and setup steps in that exercise to be complete first.

The goal of this exercise is to get an access token using the authorization code flow and PKCE. This exercise will walk you through the flow manually without writing any code. You are of course free to write code to do this instead if you'd like, but the instructions here will show you the step by step process of what's happening under the hood.

From the side menu of your Okta Developer dashboard, click on **Applications** and choose **Applications**.

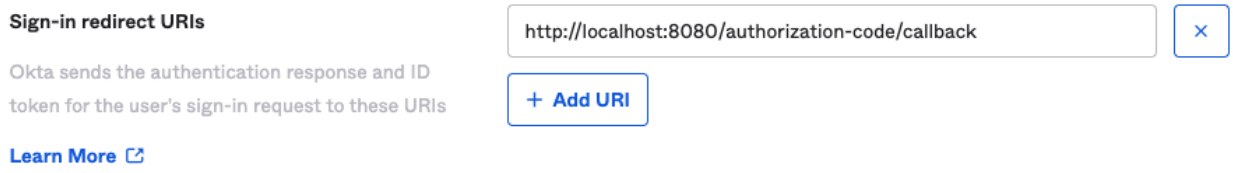


Click **Create App Integration**, then in the popup dialog choose **Web** as the platform and **OpenID Connect** as the sign-on method.



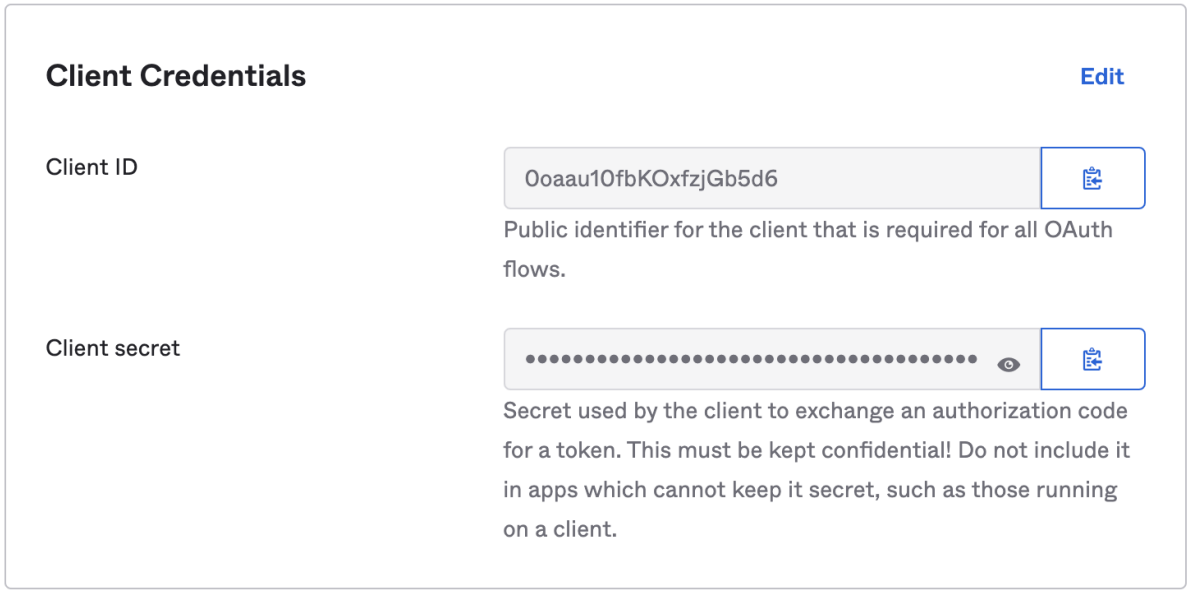
On the next screen, we'll need to add a Sign-in redirect URL. This is where the OAuth server will send the user back to after they log in. For this exercise, we'll use a placeholder URL that will help us out.

Under **Sign-in redirect URIs**, replace the default value with https://example-app.com/redirect as the redirect URI for your application



You can leave the rest of the settings at their defaults. Click **Save** to finish creating the app.

Once you've created the application, you'll get a client ID and secret. You'll need these to complete the OAuth flow.



With your application credentials in hand, you're ready to start the flow! To do that, you'll need to use the authorization server's authorization endpoint that you found in the Introduction exercise. Look up the URL from your notes or copy it from the Introduction exercise on oauth.school.

Before you can create the authorization URL, you need to create the PKCE Code Verifier. Generate a random string between 43-128 characters long. You can do this on your own, or use this web page to generate a random string for you:

https://example-app.com/pkce

Next, you need to create the Code Challenge, which is the Base64-URL-encoded SHA256 hash of the random string you generated previously. You can write code to do this yourself, or you can use the same link above to calculate it for you.

Save the Code Verifier and keep it secret, you won't need that until the end.

You are ready to build the URL to send the user to go log in. You'll start with the authorization endpoint, and add a bunch of query string parameters describing your request.

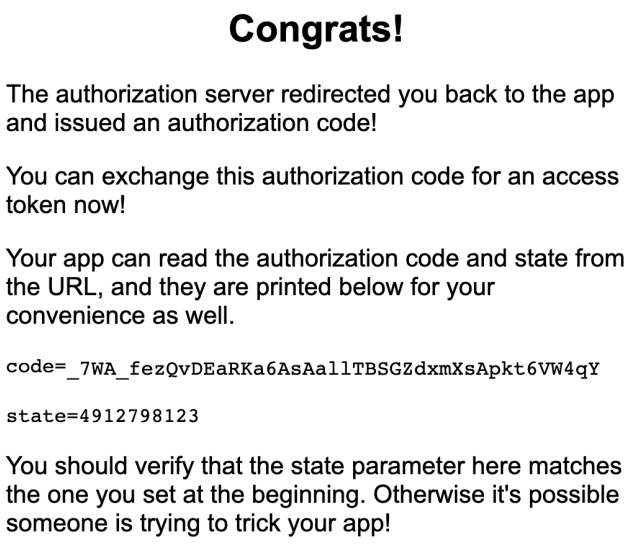
Fill in the placeholder values with your own values. (Make sure to replace the curly brackets, those are just to indicate placeholder values. Also, sometimes copying from a PDF causes extra funny characters to appear and breaks the command, so if you are having trouble, try re-typing everything from scratch.)

```
https://dev-xxxxxx.okta.com/oauth2/default/v1/authorize?
  response_type=code&
  scope={YOUR_SCOPE}&
  client_id={YOUR_CLIENT_ID}&
  state={RANDOM_STRING}&
  redirect_uri=https://example-app.com/redirect&
  code_challenge={YOUR_CODE_CHALLENGE}&
  code_challenge_method=S256
```

Paste the completed URL into the OAuth for Web Applications exercise to check your work. The tool will take a quick look and let you know if you've forgotten any parameters or if any of them look obviously wrong. The real test will be clicking the "Log In" link to see if the authorization server accepts your request! After you click "Check Your URL", the button will turn in to "Log In", and if you look at the link on the button, it's exactly what you typed in to the box. Click that link and it will open a new window to the authorization server.

Since you're already logged in to your Okta account, you should be redirected back immediately. If you want to see what happens for a new user, open that URL in an incognito window instead.

You'll be redirected back to the example-app.com redirect URL. Look up in the query string to see if you got back and authorization code or if there is an error. If there is an error, the message will help you figure out what is wrong. It could be that you included a scope that doesn't exist, or the client_id is wrong.



If you got back an authorization code, you're ready to exchange that for an access token!

Now you'll need to make a POST request to the token endpoint to exchange that temporary authorization code for an access token. You can do this manually with curl or Postman, or you can write code for it as well. You'll need the complete response from this request in order to check your work.

If you're using curl, you can start with the example command below and replace the values with your own. (Make sure to replace the curly brackets, those are just to indicate placeholder values. Also, sometimes copying from a PDF causes extra funny characters to appear in the terminal and breaks the command, so if you are having trouble, try re-typing everything from scratch.)

```
curl -X POST https://dev-xxxxxx.okta.com/oauth2/default/v1/token \
  -d grant_type=authorization_code \
  -d redirect_uri=https://example-app.com/redirect \
  -d client_id={YOUR_CLIENT_ID} \
  -d client_secret={YOUR_CLIENT_SECRET} \
  -d code_verifier={YOUR_CODE_VERIFIER} \
  -d code={YOUR_AUTHORIZATION_CODE}
```

If everything worked, you'll get a response that includes an access token! The most likely way this will fail is if you took too long between getting the authorization code and making this request. These codes last for a very short amount of time, so you'll want to get everything ready to make this POST request as soon as you can after getting the authorization code.

Paste the entire response (not just the access token) into oauth.school to check your work! If everything worked, you'll get a message saying you've finished!